EXPRESS MAIL LABEL NO. EV 384 569 018 US
MARCH 19, 2004

TITLE

METHOD AND SYSTEMS FOR PROVIDING DATA REPLAY, REPROCESS AND RETRIGGER FUNCTIONS IN AN ANALYZER

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This patent application claims priority and benefit under 35 U.S.C. § 119 to United States Provisional Patent Application Serial No. 60/456,009 filed on March 19, 2003, and that patent application is hereby incorporated by reference in its entirety.

BACKGROUND

[0002] This disclosure relates to capturing data traffic on a bus or other transmission medium and analyzing that data. Various techniques are disclosed.

SUMMARY

[0003] Data travelling on a bus or other transmission medium may be captured for analysis. As part of that analysis, captured data may be stored in a memory location from which it may be replayed or reprocessed as desired. A retrigger function may be included. Such a system may be called a "Replay Analyzer" for convenience of reference in this document. A Replay Analyzer is able to perform post-capture analysis using much of the same powerful logic and circuitry used during the capture for triggering, monitoring, and statistics.

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] Figure 1 is a block diagram of an example Replay Analyzer showing the main interconnections between the blocks and showing the data flow for input data and replay data.

[0005] Figure 2 is a block diagram of example term logic of the example Replay Analyzer showing the inputs and outputs. Figure 2 also shows the Event Statistic Counters.

[0006] Figure 3 shows the detail of example Term Logic in a Replay Analyzer.

[0007] Figure 4 is a simpler alternative implementation of the Replay Analyzer which does not include the Selective Capture Logic, and does not include the logic associated with correctly timing the replay.

[0008] Figure 5 shows example Selective Capture logic which generates the STORE signal for a Replay Analyzer.

[0009] Figure 6 shows the 12 Levels of the Trigger Mesh and the possible paths from each level to other levels in an example Replay Analyzer.

[0010] Figure 7 is an example flow diagram of eight term levels, the two timer levels, and the two counter levels in the Trigger Sequencer of a Replay Analyzer.

[0011] Figure 8 is an block diagram of example Replay Logic showing the inputs and outputs of each block within the Replay Logic in a Replay Analyzer.

[0012] Figure 9 is a Block Diagram of example Trace Buffer Control Logic and the Trace Buffer, showing the inputs and outputs of each block in a Replay Analyzer.

[0013] Figure 10 is example detail of the Stop Logic function in a Replay Analyzer.

DETAILED DESCRIPTION

[0014] The example implementation and features of a Replay Analyzer discussed herein will be easier to understand with some background on possible benefits that may be achieved through use of a Replay Analyzer. Unlike general-purpose analyzers (e.g. logic analyzers), bus-specific protocol analyzers often provide additional features beyond capture-and-display capability. Those additional features may include: 1) real-time protocol

monitoring, 2) real-time statistical analysis, and 3) traffic generation capabilities. Some protocol analyzers may also add additional value to the traffic generation capability by providing methods for replaying the captured traffic during traffic generation mode.

[0015] The Replay Analyzer brings these features to a general-purpose analyzer which can be used for any bus or transmission medium, while simultaneously speeding up the post-capture processing of trace data by replaying and processing it at high-speed. The Replay Analyzer also decreases the complexity of the user interface (increasing the ease-of-use) by ensuring a significant commonality in the user interface for controlling the normally separate functions of real-time monitoring, off-line analyzing, and generating traffic. This may be achieved through new hardware techniques which make it possible to share the hardware resources used for real-time monitoring and statistics, post-capture analyzing, and replaying saved traffic. This commonality provides the additional benefit of decreasing the system cost by increasing the utilization of each hardware function.

High-Level Functions:

[0016] Figure 1 depicts a block diagram 100 of an example Replay Analyzer. As shown in Figure 1, the primary blocks in the Replay Analyzer are the Data Input Port 101, Trace Buffer 102, Term Logic 103, Event Statistic Counters 104, Selective Capture Logic 105, Trigger 106, Replay Logic 107, Replay Output Port 108, Timestamp Upcounter 109, and Control Port 110. Those familiar with protocol analyzers, monitors, and traffic generators, will recognize that the Replay Analyzer eliminates a number of logic blocks typically used to provide the analyzer's feature set. For example to generate traffic, no separate buffer or state machine is required for generated packets. For fast trace statistics, there are no separate statistics counters, terms, or content-addressable memory, since the Event Statistic Counter 104 hardware performs this same function on the trace by gathering replay statistics. Hardware search logic is not required due to the replay analyzers ability to use the trigger for fast event gathering operations. For fast data gathering or filtering, the replay statistics counters or replay trigger can be

used to identify specific data values or events. For very fast creation of a Trace Histogram, or for locating commands, the replay statistics and replay trigger functions can be used to gather histogram and command data.

[0017] The Replay Analyzer may be coupled to a transmission medium (bus) through an adapter pod. Using an adapter pod, almost any bus can be analyzed or emulated (replayed) including wired, wireless, and optical mediums, as long as the analyzer has sufficient bandwidth. Although separate input and output pods may used, it is also straightforward to use a single pod with bi-directional I/O channels, instead of an Input pod for the Input Channels, and an Output pod for the Output Channels.

[0018] The analyzer's bandwidth, and ability to support busses of the future, is a function of the number of channels used in the input port, the sample rate, and the speed that the input channels can be processed at. The versatility of being able to support various busses is made possible through the combination of the adapter pod, input channels, and the Term Logic. The Term Logic performs pattern recognition for the other analyzer functions that process incoming or outgoing data. For example, the Event Statistic Counters provide long-term statistics regarding the types of events occurring, with each event type being defined by a term. Term events can be commands, packets, addresses, data transfers, signal assertions, or any other occurrence that can be defined as a pattern on the incoming channels.

[0019] The Selective Capture logic uses terms from the Term Logic to capture only incoming activity that matches predefined or user defined patterns, thus ensuring the Trace Buffer is utilized to store only useful data, and filter everything else.

[0020] The Trigger Sequencer is used to trigger on an event or sequence of events and eventually stop the Trace Buffer Controller from continuing to increment the Trace Buffer address and write (during capture) or read (during replay) the Trace Buffer memory. Whether the Trace is being written or read

depends on whether the analyzer is in Capture Mode or Replay mode. The time between the trigger and when the capture or replay is halted depends on the value in the Post-Trigger Counter.

[0021] The Replay Logic provides a means for selecting the data flow source and direction, either from the Data Input Port and into the Trace Buffer and trigger or trigger Term Logic, or out of the Trace Buffer and into the trigger or trigger Term Logic. The Replay Logic also optionally assists in ensuring that each trace event is replayed with the same timing it was captured with.

[0022] The Replay Output Port provides the replayed trace data to an Output Adapter Pod, which provides the physical connection to the bus, and any serialization and signaling conversion necessary, depending on whether the bus medium is analog or digital, wired or wireless, optical or electrical, single-ended or differential. The Output Adapter Pod generally performs the inverse functions of the Input Adapter Pod, taking the replayed data from the Trace Buffer and converting it back to the original format that was presented to the Input Adapter Pod..

[0023] The Control Port provides a bus structure whereby a local or remote user or processor can configure the replay analyzer logic.

[0024] By providing the above structure and functionality, the Replay Analyzer permits a user to capture data moving across a bus and replay it through the analyzer multiple times to look for desired data patterns and to perform other analysis of the data. The replay function permits construction of a far simpler analyzer hardware than would be used if either a hardware or software search function were implemented.

Term Logic:

[0025] Figure 2 depicts a block diagram 200 of example term logic of the example Replay Analyzer showing the inputs and outputs and the Event Statistic

Counters 203 through Term Logic 204. As shown in Figure 2, the terms are used to perform event pattern recognition on the I/O channels. The example Replay Analyzer has 108 I/O channels 201, which when combined with the 32 Timestamp bits, StatesA-D and StatesE-H bits, TRG bit, and TS_MAX bit, add up to 144 bits, which is a width that maps into two 72-bit SDRAMS. Other channel width configurations may be a better fit for different buffer implementations. In an example implementation, the terms can mask any I/O channel from comparison, or compare any channel against an expected edge or level. Each term can be configurable to compare each I/O channel against any of the following possibilities – High, Low, Rising Edge, Falling Edge, Either Edge or DontCare. Also, each I/O channel of the term can be configured to perform compares of levels (High or Low) using either the current or previous value of the I/O channel. This makes it possible to configure a term so that it compares the levels during the setup time or hold time with respect to the edge comparisons.

[0026] Figure 3 shows the detail of example Term Logic in a Replay Analyzer. As shown in Figure 3, the Term Logic 301 can use all the incoming channels as inputs during capture. During replay operations Term Logic can use the replay channels as well as some replay status signals such as Replay_TRG, which is asserted once the Trigger event occurs, or Replay_TS_MAX which indicates a Timestamp rollover, or Replay_StatesA-D and Replay_StatesE-H which indicate which set of states caused the capture, where each set of states is defined for a different purpose, such as capturing different directions of a bi-directional bus. During initial capture of bus data, only the non-replay signals are used to define the term, and the replay signals are configured to the DontCare condition. During replay operations the replay signals aid in processing functions, for example Replay_TS_MAX is used in determining the total length of time taken by the trace, by counting the number of times the timestamp rolled over. Replay_TRG indicates where the trace triggered during capture. Replay_StatesA-D and Replay_StatesE-H indicate what type of event caused the capture, for example, if StatesA-D are defined as transfers on the buss's transmit direction, and StatesE-H are defined as the transfers in the receive direction, then the Replay_StatesA-D

signal will be asserted only for valid transfers in the transmit direction, thus making it possible to define replay terms that are only asserted for valid transfers in the desired direction, and not asserted if the transfer is in the opposite direction, or not valid for some other reason such as a timestamp rollover (TS_MAX), or a timing mode capture of a channel transition at a different time than during the valid data transfer.

[0027] As shown in Figure 3, a term is the assertion of a term signal, where the term signal represents a match between the I/O channels and the programmed compare value for the term. All terms except the Timing Term use a logical AND of the individual I/O channels to create the final term signal. The Timing Term is asserted when an edge occurs on any of the I/O channels, so the timing term uses a logical OR of the I/O channels, and each channel input to the Timing Term is typically configured to be true when either edge (rising or falling) appears on the channel. For all terms, the "D" flip-flop on the input of each term channel is used to provide a delayed version of the incoming signal, making it possible for the term to detect an edge by detecting that the current signal level is different than it was previously.

Adapter Pod(s):

[0028] An adapter pod 110 of Figure 1 can be used to interface the Replay Analyzer to the physical characteristics of the bus being analyzed or transmitted on. An Input Adapter Pod is used for analyzing, and an Output Adapter Pod is used for transmitting. Each analyzer channel can include a differential signal, and cabling can be used to connect to the pod. Or a single ended signal can be used for each channel, with the pod connecting directly to the analyzer, thus eliminating the loss and crosstalk associated with cables. The pod performs the signal conversion from the differential signaling used by the analyzer, to the method used by the bus, which may be analog or digital, single-ended or differential, parallel or serial, wired or wireless or optical. Each method has its own special transceivers and adaptive requirements for the transmission medium being used. For example, gigabit serial busses

typically require that the analyzers input channels be deserialized using a PHY or SERDES, and the output channels must be serialized again using a PHY or Serializer. Techniques for serializing and deserializing are commonly known in the art of gigabit busses. For optical busses, special optical transceivers are typically employed, which may or may not have the SERDES built into the optical device. Fibre Optic cable is the typical medium for optical transmission lines. For wireless mediums, transmitters and receivers are employed in the adapter pods using suitable antennas as the connection to the medium. For analog transmissions Analog-to-Digital Converters (ADC's) are typically used to convert the analog signal to the parallel digital transmission of the Replay Analyzer. In the output direction, Digital-to-Analog Converters (DAC's) are used. These methods of converting a parallel digital electrical bus to a non-copper or non-parallel or non-digital or non-electrical medium are well known in the art.

[0029] In addition to converting the bus signaling to the parallel differential format, the Input Adapter pod for many busses includes logic which creates special signals that aid the decoding, triggering, filtering, and statistics functions of the analyzer, by making it possible for a single term to recognize that a specific captured event has special meaning due to its position in a sequence of events. For example, serial busses often transmit packets of information, and the Input Adapter Pod will deserialize the packets into 32-bit chunks, called Double-Words, or D-words. In order to process the information in each D-word, it is useful for each D-word to be transmitted to the analyzer with accompanying count bits that indicate which D-word in the packet this D-word is. These count bits are created by the Input Adapter pod and sent with the D-word to the analyzer so that the terms can use the count bits to know which D-word in the packet it is. For example, if the user wants to define a Term as being a specific value in the Address Field of the Packet Header, the count bits can be used to identify which D-Word is the Address Field. In a similar manner, the Input Adapter pod can create protocol specific meanings for the Input channels such as "Command" or "Status" or "Error". Groups of input channels can be created by the Input Adapter pod, for example a 2-bit group called "Error" may use a 2-bit pattern of 00 to mean

No Error, a pattern of 01 means Coding Error, 10 means Disparity Error, and a pattern of 11 means Primitive Error. This group makes it possible for the logic in the pod to decode the 8b/10b coding used by most gigabit busses and provide signals to the analyzer which help it trigger on specific types of bit or byte error conditions. Generally, the Input Adapter pod is responsible for providing any bus-specific decoding of the bus signaling that must occur in real-time. For triggering purposes, the trigger sequencer can also be used to trigger on sequences of events that aren't decoded by the pod, which significantly increases the decoding power of the analyzer during both Capture and Replay operations.

Selective Capture Logic:

[0030] Capturing traces can be done in State Mode, Transitional Timing mode, Fixed Frequency mode, or otherwise as desired. State Mode and Transitional Timing mode both rely on a Timestamp for each captured event in order to display or replay the data with the correct timing. Fixed Frequency mode performs a capture on each clock cycle of the 250MHz sample clock, thus no timestamp is required to display or replay the data with correct timing. Figure 4 shows a simpler alternative implementation of a Replay Analyzer 401 with a reduced number of blocks required to perform replay operations, since the data is captured at a fixed frequency. It includes replay logic 402, trigger logic 403, trace buffer control logic 404 and trace buffer 405. Note that patterns in the trace buffer (which represent patterns found in bus traffic) are found by replaying the contents of the trace buffer and analyzing them with trigger logic 403 rather than by a search function.

[0031] Referring now to Figure 5, example details of a Trigger Logic implementation are depicted via a logic diagram 501. Trigger Logic asserts a trigger signal when the data presented to it matches a predefined pattern or sequence. Fixed Frequency mode can be implemented by selecting the State Mode of capture, and then configuring one of the state terms as all DontCare values. This causes the STORE signal generated by the Selective Capture Logic to always be true, and the Trace Buffer Controller will then store every 4ns

(250MHz) sample. The Timestamp Upcounter is a 32-bit counter that rolls over to 00000000h when it reaches the FFFFFFFFh terminal count. The Timestamp Upcounter is initially reset to zero on the rising edge of RUN by the RUNRISE signal. A signal called TS_MAX (short for TimeStamp Max) is asserted on the rollover to 00000000h. Since TS_MAX causes a capture in any of the capture modes, and since TS_MAX is stored in the Trace Buffer, it is easy for software to show much longer timestamps than the 32-bit counter can count by itself, by counting the number of rollovers that have occurred in the trace up to any specific event. The Replay Trigger can be used to count these TS_MAX occurrences quickly since Replay_TS_MAX (the version of TS_MAX which is output by the buffer during replay) is an input to all Terms. At the beginning of a capture or replay, the Timestamp Upcounter is reset to 00000000h when the RUN signal is first asserted, and from then until the end of the trace it counts every cycle of the 250MHz sample clock.

[0032] The State Mode of capture can use the Selective Capture Logic in Figure 5, to identify and capture only the bus traffic that matches the patterns loaded into the state terms. In State Mode, the eight state terms (A thru H) are ORed together to generate the STORE signal, thus a STORE occurs for each clock cycle that a state term is true. The eight state terms are grouped into two groups, called StatesA-D, and StatesE-H. By configuring each group of state terms to match only the data corresponding to a single direction of a bi-directional bus, the Replay Analyzer can capture both directions simultaneously, using an Input Adapter pod which uses 54 of the 108 input channels for one direction, and the other 54 input channels for the other direction. Two signals are stored in the Trace Buffer along with the input channels called StatesA-D and StatesE-H. Since the first four states (A thru D) are used to capture traffic in one direction, the StatesA-D signal, which is an OR of the first four state terms, will only be asserted in the Trace Buffer when the 54 channels corresponding to States A-D contain valid data. The 54 channels for States A-D could be invalid if it was States E-H that caused the capture. If both the StatesA-D and StatesE-H signals are valid for the same event in buffer memory, then valid data was transferred in both directions simultaneously on the bus. Each of the 8 state terms can be

configured to qualify its channels either before the edge, or after the edge of the signal used to strobe the data into the analyzer. Since all 108 channels can be configured as an edge, any of the 108 channels can be used as the data strobe. Typically, only one signal will be used as a strobe and the rest will be configured as levels, or as DontCare signals.

[0033] The Timing Mode of capture uses the Timing Term instead of the State Terms to generate the STORE signal. The Timing Term is configured to go true whenever an edge occurs on any of the input channels. An edge is defined as any input where the level of the input during the previous sample is different than the current sample. Since each event stored is timestamped, it is easy for software to reconstruct a timing waveform of the signal activity from the Timing Mode trace data, which is especially useful for troubleshooting bus timing problems such as setup or hold time violations.

Stopping the Trace or Replay:

[0034] Example Stop Logic 1001 is shown in Figure 10. The Stop Logic generates the DONE signal which tells the Trace Buffer Controller to halt a capture or replay operation. The Replay Analyzer can be configured to stop capturing when the Trace Buffer is full. This configuration is performed by the host by writing a register which sets the EN_FULLMODE bit. During a capture or replay in this mode, DONE will be asserted when the LATCHED_ALMOST_FULL signal is asserted by the Trace Buffer Controller. Capture and Replay can also be configured to loop on the Trace Buffer until the operation is stopped by the user in Manual Stop mode, or by the Trigger in Stop After Trigger mode. In Stop After Trigger mode, a Post Trigger Counter is used to stop the capture or replay some number of events after the trigger occurs. The Post Trigger Counter is loaded by the host when the user selects the ratio of Pre-Trigger and Post-Trigger data he wants to see in the buffer once the capture is complete. Typically, in replay mode the Post Trigger Counter is loaded such that the replay stops only after the desired number of addresses corresponding to a trigger match have been latched into the TRG ADDRESS FIFO. The assertion of the Post Trigger Counter

output is latched and then generates the DONE signal by being ANDed with the EN_TRIGMODE signal, which is from a register written by the host, and ANDed with the LATCHED_ALMOST_FULL signal, which indicates the buffer has reached the end or wrapped at least once during the operation. The LATCHED_ALMOST_FULL signal ensures that the entire trace buffer is utilized during a capture even if the Trigger occurs before the desired amount of Pre-Trigger data is captured. During replay, the LATCHED_ALMOST_FULL signal is forced high so the replay can be stopped without reaching the end of the buffer or wrapping. In all modes, the user can stop the capture manually by pressing the STOP button which causes the host to simultaneously deassert EN_FULLMODE, EN_TRIGMODE, and RUN, which asserts the DONE signal. When DONE is asserted, the Trace Buffer Controller disables further writing or address incrementing. During trace capture the LATCHED_ALMOST_FULL signal is generated by latching the output of a comparator which compares the current Trace Buffer address with a LAST_ADDRESS register which gives the stop logic sufficient time to stop before the Trace Buffer address counter wraps to the beginning again. In all modes, once the trace is stopped, host software reads the current address value to determine which address is the last captured address in the trace. In Stop After Trigger mode, when the Trigger signal is asserted a post-trigger upcounter is enabled. When the post-trigger upcounter overflows, the DONE signal is asserted, disabling further writing or address incrementing.

The Trigger Sequencer:

[0035] An embodiment of the Trigger Logic uses a Trigger Sequencer. The Trigger Sequencer can use a state-machine architecture, capable of changing states on a single clock cycle. One additional feature of the Trigger Sequencer state machine is that up to 12 states (also called levels) can be enabled simultaneously. As shown by example in Figure 6, each level of the state machine 6001 can enable any or all of the other levels upon exiting the level. Figure 7 is an example flow diagram of eight term levels, the two timer levels, and the two counter levels in the Trigger Sequencer of a Replay

Analyzer. Each of the 12 levels in the Trigger Sequencer consists of a term or count that it waits for as its input, and a 13-bit "THEN" register which is programmed with up to 12 states (levels) that will be enabled once the input becomes true, and is also programmed to generate the TRG signal (1-bit). This architecture allows the user to program the sequencer using a familiar If/Then/ElseIf structure. Each Level can be enabled singly, or enabled at the same time as any or all of the other levels. This ability to have multiple levels enabled at one time makes it possible to create ElseIf structures. When started, the trigger sequencer has Level 1 enabled. When the event which Level 1 is waiting for occurs, the sequencer enables the next level(s) using the "THEN" register contents. If a single bus event satisfies the input conditions of multiple enabled levels, the "THEN" registers for all satisfied levels are used to enable the next levels. Eight of the levels simply wait for their specific term to become true. Two levels are Counter levels which wait for their term-counter to reach its terminal count. Two levels are Timer Levels which wait for their time-counter to reach its terminal count. During Capture, the TRG signal is stored in the Buffer so that the trigger position can easily be identified.

Event Statistic Counters:

[0036] The example Replay Analyzer has six Event Statistic Counters 203 as indicated in Figure 2. These counters are incremented each clock cycle that the input term feeding the counter is true. The counters can be reset individually by the user interface, and will roll over once the maximum count is reached. In order to maintain larger counts than the number of bits in the counter, host software must read these counters more frequently than the shortest possible rollover period. By reading the counters this frequently, host software can ensure every counter rollover is counted. Although the preferred embodiment uses terms to perform the pattern matching, it is anticipated that an alternate implementation would compare the incoming data using a Content Addressable Memory (CAM) device, or a sequence of terms, instead of a single term, and the Event Statistic Counters would be incremented by the FOUND output signal of the CAM device. Ternary CAM

devices allow for inclusion of Dontcare values in the comparison. Such architecture makes it possible to keep statistics on larger sequences of events without creating special input channels with the adapter pod. Using a CAM instead of a term, up to the word size of the CAM can be used to compare packet data. During Capture, the Event Statistic Counters gather real-time statistics from the bus. During replay, the Event Statistic Counters are used to gather statistics on trace data very quickly for up to six different events. The terms (or CAM or sequence) can then be changed and an entirely different set of statistics can be gathered on the same trace data by replaying again.

Replay Logic:

[0037] As shown in Figure 8, the Replay Logic 107 (initially depicted in Figure 1) selects whether the source of data presented to all internal analyzer functions is the Input Adapter Pod, or the Trace Buffer. Replaying the captured data is useful for generating traffic, or for high-speed processing, analyzing, and sorting of the trace data. If the replay is for traffic generation purposes, the Replay Output Port and Output Adapter Pod are enabled during the replay. Otherwise, the Replay Output Port and Output Adapter Pod are disabled. A Replay operation can replay the entire trace or any portion of it. The starting address is controlled by loading the Address Counter in the Trace Buffer Controller with the desired starting address. The stopping address is controlled by the value loaded into the LAST_ADDRESS register. The Last Address register is the same register used to stop a trace capture when the stop mode is configured for Stop When Buffer Full. Similar to a trace capture, the replay operation can be stopped manually at any time by asserting the DONE signal. During replay processing of trace data, the trigger can be used to stop the replay after an event or event sequence, similar to a trace capture.

[0038] Figure 9 is a Block Diagram 901 of example Trace Buffer Control Logic 902 and the Trace Buffer 903, showing the inputs and outputs of each block in a Replay Analyzer. As shown in Figure 9, the TRG ADDRESS FIFO 907

latches the address value on the output of the Replay FIFO 905 when the TRG signal is asserted. This latched trigger address is used by software during or after a replay operation to determine which replayed addresses resulted in a trigger. There is sufficient pipelining before the TRG ADDRESS FIFO to ensure it latches the correct address when TRG is asserted. The TRG ADDRESS FIFO asserts the Not_Full output to continue loading events during the replay. Not_Full is used to indicate that there is more room for trigger addresses in the TRG ADDRESS FIFO. The host can select from two sources to assert the Load_Next signal during the Replay. By asserting the En_Timing_Ctrl signal, the host selects the Timestamp Match signal from the Timing Control Comparator as the Load_Next signal. Otherwise, the Not_Full signal from the TRG ADDRESS FIFO is chosen. The source of Load_Next will be the Timestamp Match if the trace is being replayed with correct timing. Otherwise it will be the Not_Full signal from the TRG ADDRESS FIFO if the trace is being replayed to sort trace data. The TRG ADDRESS FIFO allows the host to continuously read the location of trigger events during replay without the delays associated with stopping and restarting a replay operation. Two significant advantages result from using a TRG ADDRESS FIFO instead of a single-address TRG ADDRESS LATCH. First, if the user defines specific events to be shown in the Command Listing or State Listing, a replay trigger operation can be used to flag these events very quickly instead of finding the events one-at-a-time. Second, the process of flagging the events in reverse if the user scrolls the filtered Command Listing or State Listing backwards becomes much faster, due to the ability to start the replay thousands or more events prior to the currently displayed location, and flag each trigger sequence occurrence in the forward direction until the currently displayed address location is reached. The TRG ADDRESS FIFO makes it possible to perform fast event flagging operations without stopping or restarting a replay operation.

[0039] An important feature of the Replay Analyzer is the ability to replay the captured traffic using the same timing it was captured with. This is especially important while replaying the captured data to the Output Adapter Pod, but is also important when replaying the trace to the Trigger Sequencer if the

trigger sequence is using the timing levels to trigger on a timing condition. Replaying the trace with correct timing is easier to accomplish if the trace buffer is implemented with SRAM, but if the trace buffer is implemented with DRAM or other memory requiring refresh to achieve a deeper buffer as is the Replay Analyzer's buffer, the occasional pauses for refresh cycles can make it difficult to replay with correct timing. In the Replay Analyzer this problem was solved by using the Replay FIFO 905 on the output of the Trace Buffer, and by driving the Trace Buffer with a slightly higher clock frequency than the original sample clock. The data is replayed out of the Replay FIFO using the same 250MHz clock that it was originally sampled with, and the length of the FIFO compensates for the refresh cycles that cause occasional pauses in the loading of the Replay FIFO. Although this works fine if the capture was performed at a fixed frequency of 250MHz, this technique is insufficient for captures done in Timing Mode since only transitions are captured, and the timing information is contained in the timestamp. In order to replay Timing Mode and State Mode captures with correct timing, the Replay Logic contains a comparator that is loaded with each replay timestamp. During the replay, the replay timestamp is compared with the value from the Timestamp Upcounter, which runs during replay operations just as it does during capture operations. When a match occurs, the Stored Activity from the Replay FIFO is placed on the Stored Activity With Correct Timing bus for output onto the Activity bus which feeds the Output Adapter Pod and Term Logic. When the match occurs, the next event is loaded from the Replay FIFO into the Replay Logic for the next comparison with the Timestamp Upcounter. Since the Replay FIFO is being emptied much slower than it is being filled by the Trace Buffer Controller, an ALMOST_FULL output tells the Trace Buffer Controller to pause loading of the Replay FIFO. The Timestamp Comparator in the Replay Logic can be disabled for replay operations where the original timing doesn't matter, such as when the data is being replayed to the trigger and the trigger is not using its Timer Levels. This significantly improves the speed of a replay trigger or replay statistics operation, allowing the Replay Analyzer to process or sort the trace data at a faster speed than it was captured with. For parallel busses, trace data which is captured in State Mode lacks the detail to produce the replayed data again using the original setup and hold

timings, since State Mode typically only saves one sample per data transfer. This limitation forces the user to use Timing Mode to capture the data if they plan to replay it with the original setup and hold timings. For serial busses, this limitation doesn't exist since the SERDES or PHY in the Output Adapter Pod handles the timing and transmits the serialized data using a Reference Clock built into the pod.

[0040] As shown in Figure 9, a MUX 906 may be used to select which of two signals will be used to read events out of the Replay FIFO. The MUX selects between the Not_Full signal from the TRG ADDRESS FIFO, or the Timestamp_Match signal from the Timing Control Comparator. The selection is performed by a configuration signal from the host called En_Timing_Ctrl.

Viewing the Trace:

[0041] The Replay Analyzer significantly speeds up the viewing and analysis process by being able to perform almost all decoding, flagging, finding, sorting, statistics, and filtering operations using the high-speed replay capability, using the same triggering and counting hardware that performs similar functions during trace capture. In addition to its ability to show a State Listing of all captured events, the Replay Analyzer can provide a Command Listing, which summarizes each command or packet. The Replay Analyzer creates the Command Listing by using the Trigger Sequencer to locate the next command by triggering on it during a replay operation. The Replay Trigger can locate a specific event or sequence of events in the trace buffer much more quickly than software, thus improving the analyzers trace processing performance.

[0042] The Replay Analyzer also gathers statistics on the trace data using a replay operation. Simple statistics such as the number of occurrences of an event, or the ratio of one event to another event can be gathered by defining the terms that feed the Event Statistic Counters, and replaying the trace. At the end of the Replay, the number of event occurrences can be read from the Event Statistic Counters. More complex statistics which involve the time

between events, or number of events per second, can be gathered using the Trigger Sequencer to sort the trace for events or event sequences. During a replay, software determines the addresses of events that match the trigger by reading the TRG ADDRESS FIFO. Once the replay is finished, software reads the data and timestamp at those trigger addresses and stores the event type and timestamp information.

[0043] The Replay Analyzer supports the ability to hide or filter events that would normally be shown in the Command Listing or State Listing. The Command and State Listings each have a Show/Hide menu that allows the user to define either the events to be shown in the listing, or the events which are hidden; whichever is easier to define. Processing a Show or Hide request from the user is quickly performed using a replay operation. Software configures the replay Start Address and starts the replay operation. The events to be shown or hidden are triggered on during the replay, causing their addresses to be latched into the TRG ADDRESS FIFO. Once the replay is finished, Software then creates the filtered display by showing only the events at addresses that were read from the TRG ADDRESS FIFO. This process can be performed on the entire trace, or only selected portions depending on how much of the buffer must be sorted to gather enough data for a single page of the State Listing or Command Listing.

Trace Buffer and Trace Buffer Control Logic:

[0044] As shown in Figure 9, the Trace Buffer Control Logic performs the addressing, writing, and reading of the Trace Buffer memory. The Trace Buffer Control Logic writes or reads the Trace Buffer at 266MHz, which provides the additional bandwidth required to be able to pause when necessary and insert a refresh cycle. During Capture operations, the Trace Buffer Control Logic clocks data out of the Capture FIFO into the Trace Buffer while generating the necessary control commands, addressing, and refresh that the SDRAMs require. The Capture FIFO provides sufficient memory that the Selective Capture Logic can continue to send STORE signals while the SDRAM is being refreshed.

[0045] During Read or Replay operations, the Trace Buffer Control Logic clocks data into the Replay FIFO, pausing occasionally for refresh cycles. The Replay FIFO is sufficiently large that the Replay Logic can continue to clock data out of the Replay FIFO at 250MHz without emptying the FIFO before the Trace Buffer Control Logic finishes a refresh and begins filling the FIFO at 266MHz again.

[0046] While systems and methods have been described and illustrated in conjunction with a number of specifics, those skilled in the art will appreciate that variations and modifications may be made without departing from the principles herein illustrated, described, and claimed. The present invention, as defined by the appended claims, may be embodied in other specific forms without departing from its spirit or essential characteristics. The examples and details described herein are to be considered in all respects as only illustrative, and not restrictive. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.